GR551x ANCS Profile Example Application

Version: 1.6

Release Date: 2020-06-30

Shenzhen Goodix Technology Co., Ltd.

Shenzhen Goodix Technology Co., Ltd.

Headquarters: 2F. & 13F., Tower B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828 FAX: +86-755-33338099

Website: www.goodix.com

Preface

Purpose

This document introduces how to use and verify an ANCS example in a GR551x SDK, to help users quickly get started with secondary development.

Audience

This document is intended for:

- GR551x user
- GR551x developer
- GR551x tester
- iOS developer
- Hobbyist developer
- Technical writer

Release Notes

This document is the fourth release of *GR551x ANCS Profile Example Application*, corresponding to GR551x SoC series.

Revision History

Version	Date	Description
1.0	2019-12-08	Initial release
1.3	2020-03-16	Updated the release time in the footers.
1.5	2020-05-30	Updated code in “Section 4.2 Interaction Process and Major Code”.
1.6	2020-06-30	• Modified the pin code in "Section 3.4 Bluetooth Connection" and "Section 4.2.1 Connection, Pairing, and Bonding". • Updated the code in "Section 4.2.4 Control Command".

Contents

1 Introduction

Apple Notification Center Service (ANCS) is applied to intelligent Bluetooth-enabled devices, such as wristbands and smart watches that connect to iOS devices. Through a Bluetooth Low Energy (Bluetooth LE) link, the Bluetooth devices can access media notifications from iOS devices, and send ANCS-related control commands to iOS devices.

This document introduces how to implement ANCS Client based on a GR551x SoC.

Before getting started, it is recommended to refer to the following documents.

Table 1-1 Reference documents

Name	Description
ANCS Specification	Provides ANCS protocols. Available at https://developer.apple.com/library/archive/documentation/CoreBluetooth/Reference/AppleNotificationCenterServiceSpecification/Introduction/Introduction.html#//apple_ref/doc/uid/TP40013460-CH2-SW1.
GR551x Developer Guide	Introduces the software/hardware and quick start guide of GR551x SoCs.
Bluetooth Core Spec v5.1	Offers official Bluetooth standards and core specification (v5.1) from Bluetooth SIG. Available at https://www.bluetooth.com/specifications/bluetooth-core-specification/.
Bluetooth GATT Spec	Provides details about Bluetooth profiles and services. Available at http://www.bluetooth.com/specifications/gatt.
J-Link/J-Trace User Guide	Provides J-Link operational instructions. Available at www.segger.com/downloads/jlink/UM08001_JLink.pdf.
Keil User Guide	Offers detailed Keil operational instructions. Available at www.keil.com/support/man/docs/uv4/.

2 Profile Overview

The ANCS Profile defines two device roles:

1. Server: iOS devices serve as the Central, providing services and data sources.
2. Client: Bluetooth devices serve as the Peripheral capable of detecting services from iOS devices (the Central) as well as reading and writing data after being connected to an iOS device.

The interaction process between the Server and the Client is presented in the figure below:

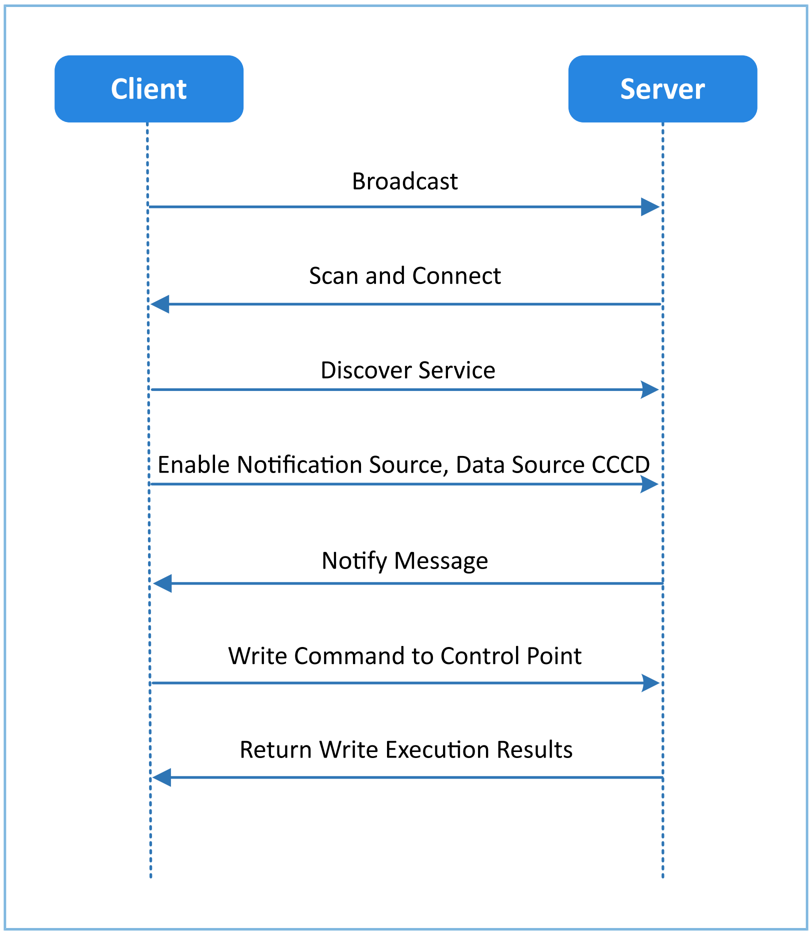

Client
Server
Broadcast
Scan and Connect
Discover Service
Enable Notification Source, Data Source CCCD
Notify Message
Write Command to Control Point
Return Write Execution Results

Figure 2-1 Client-Server interaction process

ANCS includes three characteristics as below.

Table 2-1 ANCS characteristics

Characteristic	UUID	Type	Support	Security	Property
Notification Source	9FBF120D-6301-42D9-8C58-25E699A21DBD	128 bits	Mandatory	None	Notify
Control Point	69D1D8F3-45E1-49A8-9821-9BBDFDAAD9D9	128 bits	Mandatory	None	Write
Data Source	22EAC6E9-24D6-4BB5-BE44-B36ACE7C7BFB	128 bits	Mandatory	None	Notify

3 Initial Operation

This chapter introduces how to rapidly verify an ANCS Client example in a GR551x SDK.

Note:

SDK_Folder is the root directory of GR551x SDK.

3.1 Preparation

Perform the following tasks before running an ANCS Client example.

- **Hardware preparation**

Table 3-1 Hardware preparation

Name	Description
J-Link debug probe	JTAG emulator launched by SEGGER. For more information, visit www.segger.com/products/debug-probes/j-link/.
Development board	GR5515 Starter Kit Board (GR5515 SK Board)
Connection cable	Micro USB 2.0 serial cable

- **Software preparation**

Table 3-2 Software preparation

Name	Description
Windows	Windows 7/Windows 10
J-Link driver	A J-Link driver. Available at www.segger.com/downloads/jlink/.
Keil MDK5	An integrated development environment (IDE). Available at www.keil.com/download/product/.
iOS device	An iOS device supporting Bluetooth LE 4.0 or later versions, such as iPhone 4s/iPad 3
GProgrammer (Windows)	A GR551x programming tool. Available in `SDK_Folder\tools\GProgrammer`.
GRUart (Windows)	A GR551x serial port debugging tool. Available in `SDK_Folder\tools\GRUart`.

3.2 Hardware Connection

Connect the GR5515 SK Board to a PC with a Micro USB 2.0 cable.

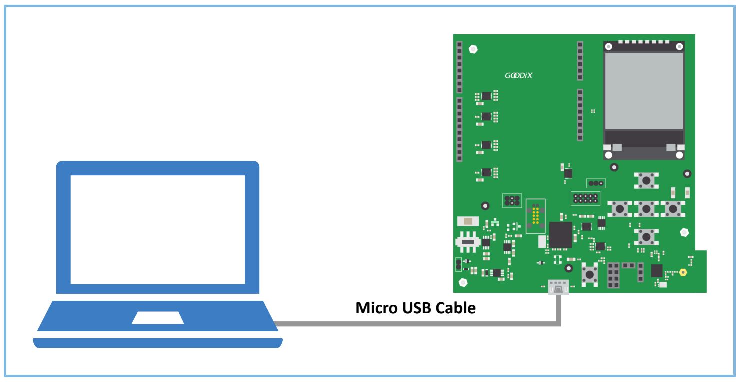


Figure 3-1 Hardware connection

3.3 Firmware Download

Download firmware of the ANCS example, *ble_app_ancs_c_fw.bin*, to a GR5515 SK Board by using GProgrammer. For specific operations, refer to *GProgrammer User Manual*.

Note:

The *ble_app_ancs_c_fw.bin* is in SDK_`Folder\projects\ble\ble_peripheral\ble_app_ancs\build`.

3.4 Bluetooth Connection

Power the GR5515 SK Board on. Turn on Bluetooth on an iOS device to scan nearby Bluetooth devices. The device discovers a GR5515 SK Board with an advertising name of **Goodix_ANCS_C**, as shown in Figure 3-2.

Note:

This document is based on tests on an iPhone 7 running on iOS 11.03. The interface can be different depending on the device and operating system in use.

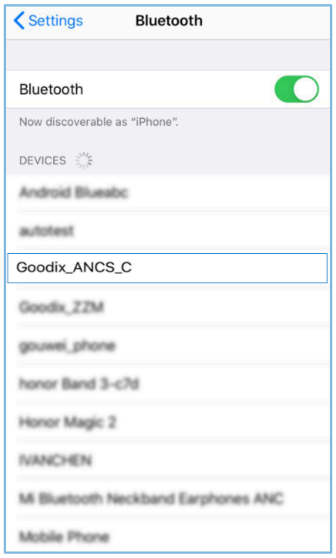


Figure 3-2 Discovering **Goodix_ANCS_C**

Tap **Goodix_ANCS_C** to connect the device to the GR5515 SK Board. As a pairing request box pops up as below, enter **888888** (see descriptions of the app_sec_rcv_enc_req_cb() function in the "Section 4.2.1 Connection, Pairing, and Bonding"), and tap **Pair**.

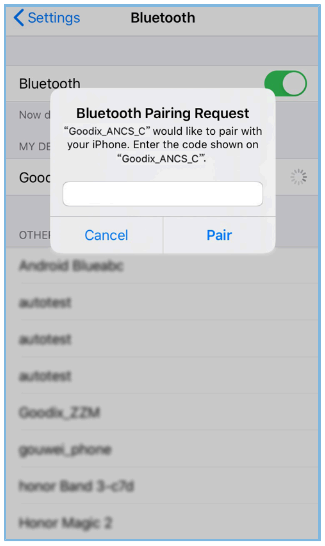

Settings
Bluetooth
Bluetooth
Bluetooth Pairing Request
"Goodix_ANCS_C" would like to pair with your iPhone. Enter the code shown on "Goodix_ANCS_C".
Cancel
Pair

Figure 3-3 Entering pairing password

After pairing, **Goodix_ANCS_C** displays as **Connected** under **MY DEVICES**.

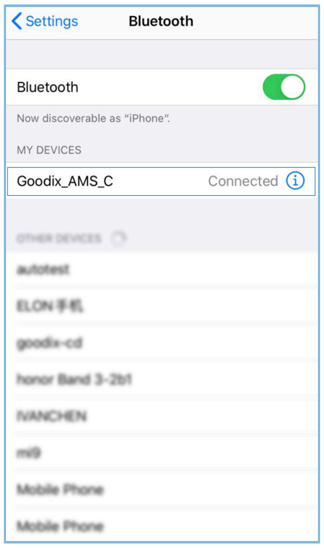


Figure 3-4 Successful pairing

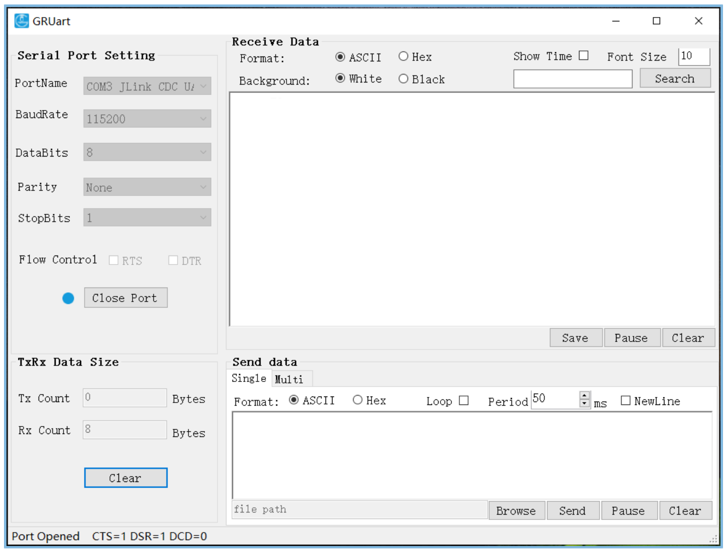
3.5 Serial Port Settings

Start GRUart, and configure serial ports according to parameters shown in the table below.

Table 3-3 Configuring serial port parameters on GRUart

PortName	BaudRate	DataBits	Parity	StopBits	Flow Control
Select on demand	115200	8	None	1	Uncheck

After configuration, click **Open Port**, to enter the interface as shown in the figure below.



```
Serial Port Setting
PortName     COM3 JLink CDC U
BaudRate     115200
DataBits     8
Parity       None
StopBits     1
Flow Control ☐ RTS  ☐ DTR
Close Port

TxRx Data Size
Tx Count  0  Bytes
Rx Count  8  Bytes

Receive Data
Format:      ◉ ASCII  ○ Hex      Show Time ☐   Font Size 10
Background:  ◉ White  ○ Black

Send data
Single  Multi
Format:  ◉ ASCII  ○ Hex   Loop ☐   Period 50 ms   ☐ NewLine

Port Opened   CTS=1 DSR=1 DCD=0
```

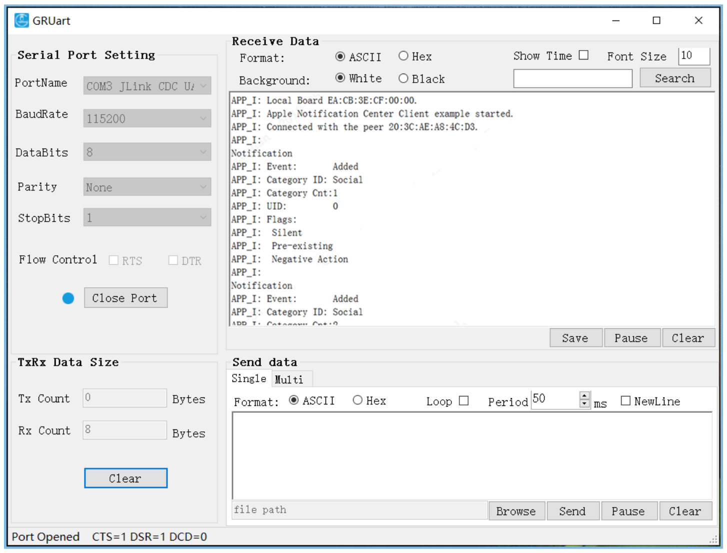
Figure 3-5 Serial port settings on GRUart

The GRUart displays no data in **Receive Data** and **Send Data** because no AMS notification is published.

3.6 Test and Verification

Users can verify whether ANCS runs normally according to serial port printing information on GRUart.

This section describes how to verify ANCS operations by taking a notification as an example, as shown in the figure below. For details, refer to *Apple Notification Center Service (ANCS) Specification*.



```
APP_I: Local Board EA:CB:3E:CF:00:00.
APP_I: Apple Notification Center Client example started.
APP_I: Connected with the peer 20:3C:AE:A8:4C:D3.
APP_I:
Notification
APP_I: Event:       Added
APP_I: Category ID: Social
APP_I: Category Cnt:1
APP_I: UID:         0
APP_I: Flags:
APP_I:  Silent
APP_I:  Pre-existing
APP_I:  Negative Action
APP_I:
Notification
APP_I: Event:       Added
APP_I: Category ID: Social
APP_I: Category Cnt:2
```

```
Serial Port Setting
PortName     COM3 JLink CDC U
BaudRate     115200
DataBits     8
Parity       None
StopBits     1
Flow Control ☐ RTS  ☐ DTR

TxRx Data Size
Tx Count  0  Bytes
Rx Count  8  Bytes

Port Opened   CTS=1 DSR=1 DCD=0
```

Figure 3-6 Serial port printing information on GRUart

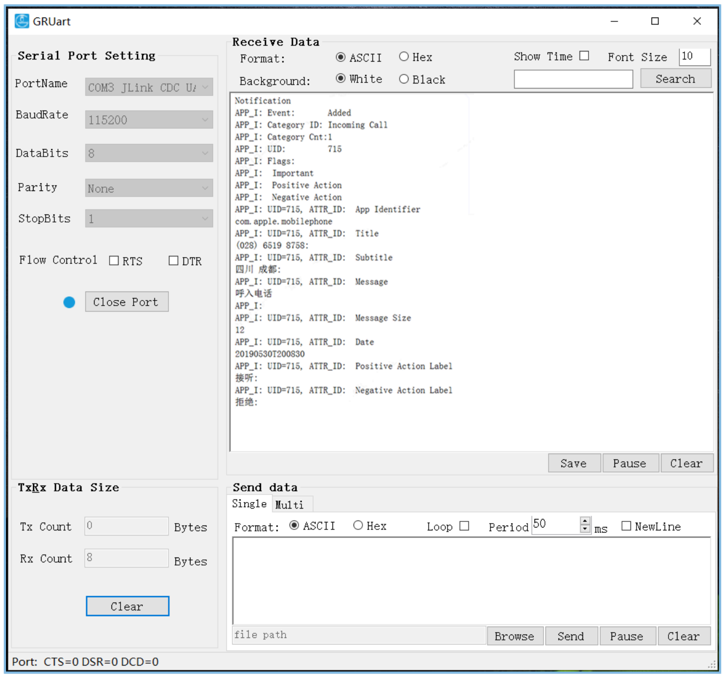
Output information from serial ports is described as below:

Table 3-4 Notification description

Name	Description
Notification	Indicates this is a notification.
Event: Added	Event type: Added
Category ID: Social	Information category: Social
Category Cnt: 1	Type quantity: 1
UID: 0	Unique identifier (UID) of the notification: 0
Flags: Silent	Information type: Silent
Pre-existing	Exists in buffer.
Negative Action	Indicates delete operation is allowed.

Take making and answering a phone call for example. Make a phone call on another phone to the tested iPhone 7. When the call is put through, the tested iPhone 7 sends a notification to the GR5515 SK Board immediately; the board then processes the received notification, and the ANCS Client example prints information on GRUart.

To check the incoming call number, users can press **OK** on the board, and the GRUart displays the following information (including the incoming call number).



PortName: COM3 JLink CDC UA
BaudRate: 115200
DataBits: 8
Parity: None
StopBits: 1
Format: [✓ ASCII] Hex
Background: [✓ White] Black
Font Size: 10
Tx Count: 0 Bytes
Rx Count: 8 Bytes
Period: 50 ms
Port: CTS=0 DSR=0 DCD=0

```
Notification
APP_I: Event:       Added
APP_I: Category ID: Incoming Call
APP_I: Category Cnt:1
APP_I: UID:         715
APP_I: Flags:
APP_I:  Important
APP_I:  Positive Action
APP_I:  Negative Action
APP_I: UID=715, ATTR_ID:  App Identifier
com.apple.mobilephone
APP_I: UID=715, ATTR_ID:  Title
(028) 6519 8758:
APP_I: UID=715, ATTR_ID:  Subtitle
四川 成都:
APP_I: UID=715, ATTR_ID:  Message
呼入电话
APP_I:
APP_I: UID=715, ATTR_ID:  Message Size
12
APP_I: UID=715, ATTR_ID:  Date
20190530T200830
APP_I: UID=715, ATTR_ID:  Positive Action Label
接听:
APP_I: UID=715, ATTR_ID:  Negative Action Label
拒绝:
```

Figure 3-7 Printed information on GRUart

To answer the phone call, press **RIGHT** on the board, then the call is connected. To decline the phone call, press **LEFT**.

Note:

For more information about buttons on a GR5515 SK Board, see “Chapter 7 Buttons and LEDs” in *GR5515 Starter Kit User Guide*.

4 Application Details

This chapter introduces the interaction process and relevant code of the ANCS Client example.

4.1 Project Directory

The source code and project file of the ANCS Client example are in `SDK_Folder\projects\ble\ble_peripheral\ble_app_ancs_c`, and the project file is in the Keil_5 folder.

Double-click the *ble_app_ancs_c.uvprojx* file, to view the ble_app_ancs_c project directory structure of the ANCS Client example in Keil. For related files, see Table 4-1.

Table 4-1 File description of ble_app_ancs_c

Group	File	Description
gr_profiles	ancs_c.c	This file implements related GATT operations in ANCS.
	ancs_protocol.c	This file contains related protocols in ANCS.
user_callback	user_gap_callback.c	This file implements GAP Callback, such as connection, disconnection, and GAP parameter update.
	user_sm_callback.c	This file implements SM Callback, such as pairing and bonding.
user_platform	user_periph_setup.c	This file configures APP LOG, device address, and power management mode.
user_app	main.c	This file contains the main() function.
	user_app.c	This file configures ANCS advertising parameters.
	user_gui.c	This file implements button operations and GUI display.

4.2 Interaction Process and Major Code

With an example of the actual interaction of ANCS Client, this section elaborates on ANCS pairing and bonding, service discovery, Client Characteristic Configuration Descriptor (CCCD) enablement, notification handling, read and write processes. Major code of each module is provided in the following sections to help users with better understanding.

The interaction process between an ANCS Server and an ANCS Client is illustrated in the figure below:

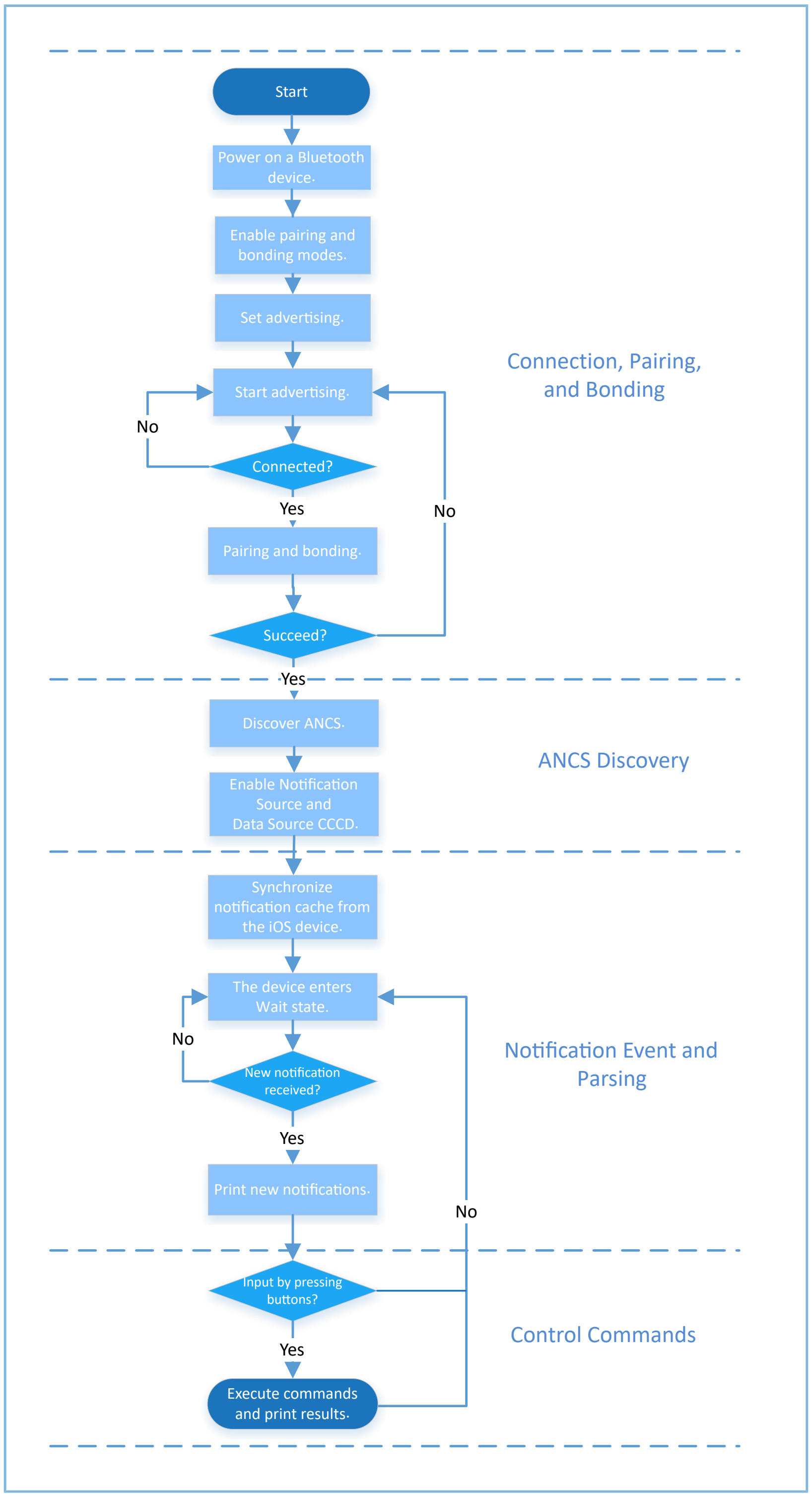

Start
Power on a Bluetooth device.
Enable pairing and bonding modes.
Set advertising.
Start advertising.
No
Connected?
Yes
No
Pairing and bonding.
Succeed?
Yes
Connection, Pairing, and Bonding
Discover ANCS.
Enable Notification Source and Data Source CCCD.
ANCS Discovery
Synchronize notification cache from the iOS device.
The device enters Wait state.
No
New notification received?
Yes
Print new notifications.
No
Notification Event and Parsing
Input by pressing buttons?
Yes
Execute commands and print results.
Control Commands

Figure 4-1 ANCS interaction process

4.2.1 Connection, Pairing, and Bonding

GR5515 SK Board begins to advertise as the Peripheral after being powered on, and the iOS device serves as the Central to search for nearby Bluetooth devices.

Turn on Bluetooth on the iOS device, and it connects to, pairs with, and bonds to the GR5515 Bluetooth device after scanning. Major code is described as below.

Path: `user_app/user_app.c` under the project directory

Name: gap_params_init();

The gap_params_init() function is used to enable pairing and bonding modes, as well as set security level (default: Security Mode Level 3; user-defined) and advertising parameters. Key code is listed as below.

Note:

For details about the settings of LE Security Mode Level, refer to "Section 10.2 LE security modes (Vol 3, Part C)" in *Bluetooth Core Spec v5.1*.

```
static void gap_params_init(void)
{
    sdk_err_t error_code;
    ble_gap_pair_enable(true);
    error_code = ble_gap_privacy_params_set(150, true);
    APP_ERROR_CHECK(error_code);

    sec_param_t sec_param =
    {
        .level      = SEC_MODE1_LEVEL3,
        .io_cap     = IO_DISPLAY_ONLY,
        .oob        = false,
        .auth       = AUTH_BOND | AUTH_MITM | AUTH_SEC_CON,
        .key_size   = 16,
        .ikey_dist = KDIST_ENCKEY | KDIST_IDKEY,
        .rkey_dist = KDIST_ENCKEY | KDIST_IDKEY,
    };
    ...
}
```

Path: `user_callback/user_sm_callback.c` under the project directory

Name: app_sec_rcv_enc_req_cb();

The app_sec_rcv_enc_req_cb() function sets custom pairing password to **888888** and assigns it to cfm_enc.data.tk.key[4] array. Code is listed as below.

```
static void app_sec_rcv_enc_req_cb(uint8_t conn_idx, sec_enc_req_t
                                   *p_enc_req)
{
    ...
    switch (p_enc_req->req_type)
    {
        ...
        // user need to input the password
        case TK_REQ:
            cfm_enc.req_type = TK_REQ;
            cfm_enc.accept = true;
            tk = 888888;
            memset(cfm_enc.data.tk.key, 0, 16);
            cfm_enc.data.tk.key[0] = (uint8_t)((tk & 0x000000FF) >> 0);
            cfm_enc.data.tk.key[1] = (uint8_t)((tk & 0x0000FF00) >> 8);
            cfm_enc.data.tk.key[2] = (uint8_t)((tk & 0x00FF0000) >> 16);
            cfm_enc.data.tk.key[3] = (uint8_t)((tk & 0xFF000000) >> 24);
            break;
        default:
            break;
    }

    ble_sec_enc_cfm(conn_idx, &cfm_enc);
}
```

Path: `user_callback/user_sm_callback.c` under the project directory

Name: app_sec_rcv_enc_ind_cb();

This function enables GR551x SDK to notify the ANCS Client example that pairing is completed. If pairing succeeds, the ANCS Client example calls a timer function with a timeout of 1 second. After a one-second delay, the example calls ancs_c_discovery_service() in the timeout response function to initiate service discovery.

Note:

When the iOS device is rebooted, the device may automatically connect to a GR5515 SK Board that it has been previously bonded to before the ANCS is enabled on the iOS device. This may result in a failure to discover the ANCS on the connection during the iOS device reboot. This is why the ANCS Client example initates service discovery one second after the paring completes.

```
static void app_sec_rcv_enc_ind_cb(uint8_t conn_idx, sec_enc_ind_t enc_ind, uint8_t auth)
{
    if (ENC_SUCCESS == enc_ind)
    {
        ancs_c_disc_delay_timer_start(conn_idx);
    }
}
```

4.2.2 Discover ANCS

After pairing with and connecting to an iOS device, the ANCS Client example initiates requests to search for ANCS of the iOS device, and enables Notify of Notification Source and Data Source according to the results of Service Browse callback function.

Path: `gr_profiles/ancs_c.c` under the project directory

Name: ancs_c_discovery_service();

This function searches for and discovers ANCS on the peer device based on a specified UUID. The scanning results are returned through the callback function, ancs_c_on_browse_svc_evt() in *user_gattc_callback.c*.

```
sdk_err_t ancs_c_discovery_service(uint8_t conn_idx)
{
    ble_uuid_t ble_ancs_uuid =
    {
        .uuid_len = BLE_ATT_UUID_128_LEN,
        .uuid = (uint8_t*)ancs_service_uuid,
    };
    return ble_gattc_prf_services_browse(s_ancs_c_env.prf_id, conn_idx, &ble_ancs_uuid);
}
```

Path: `gr_profiles/ancs_c` under the project directory

Name: ancs_c_srvc_browse_cb();

This function is used to access ANCS from the iOS device and enumerate corresponding handles of each service attribute by using UUID.

```
void ancs_c_srvc_browse_cb (uint8_t conn_idx, const ble_gattc_browse_srvc_t *p_browse_srvc)
{
    ...
    for(i=0;i<(p_browse_srvc->end_hdl- p_browse_srvc->start_hdl); i++)
    {
        if(p_browse_srvc->info[i].attr_type == BLE_GATTC_BROWSE_ATTR_VAL)
        {
        ...
        //find ancs notification source characteristic
        //find ancs control point characteristic
        //find ancs data source characteristic
        ...
        }
        else if((p_browse_srvc->info[i].attr_type == BLE_GATTC_BROWSE_ATTR_DESC) &&   //find
 cccd
                  ((*(uint16_t *)(p_browse_srvc->info[i].attr.uuid)) ==
 BLE_ATT_DESC_CLIENT_CHAR_CFG))
        {
        ...
// find notification source cccd
        // find data source cccd
        ...
        }
    }
    ...
}
```

4.2.3 Notification Event and Interpretation

When generating notifications, the iOS device sends the information to the ANCS Client, and bonds to relevant application information. The ANCS Client example is in standby state, ready to receive the latest notifications. Major code is described as below:

Path: `gr_profiles/ancs_c.c` under the project directory

Names: ancs_c_ntf_source_notify_set(); ancs_c_data_source_notify_set();

When the application status is set to ANCS_CCCD_ENABLE, this function enables the Notify properties of Notification Source and Data Source via handles obtained during service discovery.

```
sdk_err_t ancs_c_ntf_source_notify_set(uint8_t conn_idx, bool is_enable)
{
    gattc_write_attr_value_t write_attr_value;
    uint16_t ntf_value = is_enable ? PRF_CLI_START_NTF : PRF_CLI_STOP_NTFIND;

    if (BLE_ATT_INVALID_HDL == s_ancs_c_env.handles.ancs_ntf_source_cccd_handle)
    {
        return BLE_ATT_ERR_INVALID_HANDLE;
    }
    write_attr_value.handle = s_ancs_c_env.handles.ancs_ntf_source_cccd_handle;
    write_attr_value.offset  = 0;
    write_attr_value.length  = 2;
    write_attr_value.p_value = (uint8_t *)&ntf_value;

    return ble_gattc_prf_write(s_ancs_c_env.prf_id, conn_idx, &write_attr_value);
}
```

```
sdk_err_t ancs_c_data_source_notify_set(uint8_t conn_idx, bool is_enable)
{
    gattc_write_attr_value_t write_attr_value;
    uint16_t ntf_value = is_enable ? PRF_CLI_START_NTF : PRF_CLI_STOP_NTFIND;

    if (BLE_ATT_INVALID_HDL == s_ancs_c_env.handles.ancs_data_source_cccd_handle)
    {
        return BLE_ATT_ERR_INVALID_HANDLE;
    }
    write_attr_value.handle  = s_ancs_c_env.handles.ancs_data_source_cccd_handle;
    write_attr_value.offset  = 0;
    write_attr_value.length  = 2;
    write_attr_value.p_value = (uint8_t *)&ntf_value;

    return ble_gattc_prf_write(s_ancs_c_env.prf_id, conn_idx, &write_attr_value);
}
```

Path: `gr_profiles/ancs_c.c` under the project directory

Name: ancs_c_att_ntf_ind_cb();

This function dispatches two types of information: Notification Source and Data Source, and simultaneously transfers the information to corresponding parsing functions respectively.

```
static void ancs_c_att_ntf_ind_cb(uint8_t conn_idx, const ble_gattc_ntf_ind_t *p_ntf_ind)
{
    ancs_c_evt_t ancs_c_evt;

    ancs_c_evt.conn_idx = conn_idx;
    ancs_c_evt.evt_type = BLE_ANCS_C_EVT_INVALID;

    if (p_ntf_ind->handle == s_ancs_c_env.handles.ancs_ntf_source_handle)
    {
        ancs_c_evt.evt_type = BLE_ANCS_C_EVT_NTF_SOURCE_RECEIVE;
        ancs_decode_notification_source(p_ntf_ind->p_value, p_ntf_ind->length);
    }
    else if (p_ntf_ind->handle == s_ancs_c_env.handles.ancs_data_source_handle)
    {
        ancs_c_evt.evt_type = BLE_ANCS_C_EVT_DATA_SOURCE_RECEIVE;
        ancs_decode_data_source(p_ntf_ind->p_value, p_ntf_ind->length);
    }

    ancs_c_evt_handler_excute(&ancs_c_evt);
}
```

Path: `gr_profiles/ancs_protocol.c` under the project directory

Name: ancs_decode_notification_source();

This function is the data parsing function of Notification Source. In this routine, the ANCS Client example prints notification information according to requirements from the ANCS protocol. For details, refer to *Apple Notification Center Service (ANCS) Specification*.

```
void ancs_decode_notification_source(uint8_t *p_data, uint16_t length)
{
    ntf_source_pdu_t *pdu = (ntf_source_pdu_t*)p_data;
    notification_content_print(pdu);

    ancs_set_uid((unsigned int) pdu->notification_uid);}
```

Path: `gr_profiles/ancs_protocol.c` under the project directory

Name: ancs_decode_data_source();

This function is the data parsing function of Data source. In this routine, the ANCS Client example prints data contents according to requirements from the ANCS protocol. For details, refer to *Apple Notification Center Service (ANCS) Specification*.

```
void ancs_decode_data_source(uint8_t *p_data, uint16_t length)
{
    //It's the beggining of a attr info.
    if (0 == s_buf_index)
    {
        memcpy(&s_attr_size, &p_data[6], 2);
        memcpy(s_attr_buf, p_data, length);
        //It's the end of the attr info, a complete attr info is already
          stored in the buffer.
        if (s_attr_size == length - 8)
        {
            ancs_notify_attr_print();
            s_buf_index = 0;
        }
        //It's not the end of the attr info.
        else
        {
            s_buf_index = length;
        }
    }

    //It isn't the beggining of a attr info.
    else
    {
        memcpy(&s_attr_buf[s_buf_index], p_data, length);
        //It's the end of the attr info, print the buffer.
        if (s_attr_size == (s_buf_index - 8) + length)
        {
            ancs_notify_attr_print();
            s_buf_index = 0;
        }
        //It's not the end of the attr info.
        else
        {
            s_buf_index = s_buf_index + length;
        }
    }
}
```

4.2.4 Control Command

Write the control information to the Control Point of ANCS Server via ANCS Client. Details about a particular notification can be retrieved from the Data Source returned from the iOS device. In following subsections, relevant control commands are described in detail.

4.2.4.1 Access Notification Attributes

Path: `gr_profiles/ancs_protocol.h` under the project directory

Name: *ancs_protocol.h*

The macro listed below helps to access alternative values of the notification attributes.

```
typedef enum
{
    ANCS_NOTIF_ATTR_ID_APP_IDENTIFIER = 0,      /**< Identifies that the
    attribute data is of an "App Identifier" type. */
    ANCS_NOTIF_ATTR_ID_TITLE,                   /**< Identifies that the
    attribute data is a "Title". */
    ANCS_NOTIF_ATTR_ID_SUBTITLE,                /**< Identifies that the
    attribute data is a "Subtitle". */
    ANCS_NOTIF_ATTR_ID_MESSAGE,                 /**< Identifies that the
    attribute data is a "Message". */
    ANCS_NOTIF_ATTR_ID_MESSAGE_SIZE,            /**< Identifies that the
    attribute data is a "Message Size". */
    ANCS_NOTIF_ATTR_ID_DATE,                    /**< Identifies that the
    attribute data is a "Date". */
    ANCS_NOTIF_ATTR_ID_POSITIVE_ACTION_LABEL,   /**< The notification has a
    "Positive action" that can be executed associated with it. */
    ANCS_NOTIF_ATTR_ID_NEGATIVE_ACTION_LABEL,   /**< The notification has a
    "Negative action" that can be executed associated with it. */
} ancs_notification_attr_t;
```

Path: `gr_profiles/ancs_protocol.c` under the project directory

Name: ancs_notify_attr_get();

This function helps to access the corresponding notification attribute according to its UID. Take an e-mail sent from the iOS device for example. The ANCS Client example is capable of inquiring the detailed contents, receiving time, and the sender of the e-mail via this function.

```
void ancs_notify_attr_get(int uid, char noti_attr)
{
    int len = 0;
    uint8_t buf[8];
    buf[0] = CTRL_POINT_GET_NTF_ATTRIBUTE;
    memcpy(&buf[1], &uid, 4);
    buf[5] = noti_attr;
    len = CFG_ANCS_ATTRIBUTE_MAXLEN;
    buf[6] = (len & 0xff);
    buf[7] = (len>>8) & 0xff;
    ancs_c_write_control_point(0, buf, 8);
}
```

4.2.4.2 Execution

Path: `gr_profiles/ancs_protocol.h` under the project directory

Name: *ancs_protocol.h*

Two alternatives are available for users on the operation for each notification: 0 indicates agreement, and 1 indicates rejection.

```
typedef enum
{
    ACTION_ID_POSITIVE = 0,                    /**< Positive action. */
    ACTION_ID_NEGATIVE                         /**< Negative action. */
} ancs_c_action_id_t;
```

Path: `gr_profiles/ancs_protocol.c` under the project directory

Name: ancs_action_perform();

This function is used to process notifications.

```
void ancs_action_perform(int uid, int action)
{
    uint8_t buf[6];
    buf[0] = CTRL_POINT_PERFORM_NTF_ACTION;
    memcpy(&buf[1], &uid, 4);
    buf[5] = action;
    ancs_c_write_control_point(0, buf, 6);
}
```

4.2.4.3 Interaction

To help users perform interaction test in ANCS Client, this example implements the button-based commands, enabling users to operate on Control Point by pressing specific buttons.

Path: `user_app/user_gui.c` under the project directory

Name: app_key_evt_handler();

The following functions provide the process in which the response is triggered by using buttons on the GR5515 SK Board. When users press specific buttons, the ANCS Client example generates corresponding interaction commands.

Functionalities of each button are presented below:

- **OK**: Obtains and prints attribute values in various types. Such attributes values include details about text messages and e-mails, and the sending time.
- **RIGHT**: Represents **Yes** or agree. When there is an incoming call, **Yes** means answering the call.
- **LEFT**: Represents **No** or decline. When there is an incoming call, **No** means declining the call.

For detailed test methods for commands, refer to *Apple Notification Center Service (ANCS) Specification*

```
void app_key_evt_handler(uint8_t key_id, app_key_click_type_t key_click_type)
{
    uint16_t uid;
    if (key_click_type == APP_KEY_SINGLE_CLICK)
    {
        if (BSP_KEY_OK_ID == key_id)
        {
            pwr_mgmt_mode_set(PMR_MGMT_IDLE_MODE);
            uid = ancs_get_uid();
            if (uid > 0)
            {
                ancs_notify_attr_get(uid, ANCS_NOTIF_ATTR_ID_TITLE);
                ancs_notify_attr_get(uid, ANCS_NOTIF_ATTR_ID_MESSAGE);
            }
        }
        else if (BSP_KEY_LEFT_ID == key_id)
        {
            APP_LOG_INFO("pressed key left");
            uid = ancs_get_uid();
            if (uid > 0)
            {
                ancs_action_perform(uid, ACTION_ID_NEGATIVE);
            }
        }
        else if (BSP_KEY_RIGHT_ID == key_id)
        {
            APP_LOG_INFO("pressed key right");
            uid = ancs_get_uid();
            if (uid > 0)
            {
                ancs_action_perform(uid, ACTION_ID_POSITIVE);
            }
        }
    }
}
```

5 FAQs

This chapter describes problems, reasons, and solutions when verifying and using the ANCS Client example.

5.1 Why Is There No Output Information from GRUart?

- Description

  No printed information displays on GRUart, or GRUart encounters garbled printing.

- Analysis

  The *ble_app_ancs_c_fw.bin* firmware is not programmed on the board correctly, or the **BaudRate** on the GRUart is incorrect, resulting in GRUart's failure to print information.

- Solution

  Confirm on GRUart, the **BaudRate** is 115200 with **DataBits** of 8, **StopBits** of 1, None **Parity**, and no **Flow Control**. Confirm the serial ports have been connected correctly.

  If there is nothing wrong with the serial port connection, redo the firmware programming, and ensure no modification has been done on the project, then directly download the firmware to the Bluetooth device using GProgrammer.

5.2 Why does an iOS Device Fail to Scan Any Bluetooth Advertising from Goodix_ANCS_C?

- Description

  An iOS device with Bluetooth enabled fails to find advertising from **Goodix_ANCS_C**.

- Analysis

  Exceptions occur in the Bluetooth antenna connection or firmware.

- Solution

  1. Check whether the Bluetooth function on the iOS device is enabled. If the Bluetooth function is enabled, check whether the antenna of the GR551x platform is connected successfully.
  2. If the Bluetooth functions properly and the connection is successful, check the hardware problem by downloading the factory default test firmware.

5.3 Why does an iOS Device Fail to Access Notification after Connection?

- Description

  After being connected to a Bluetooth device, the iOS device cannot receive any notification.

- Analysis

  The Bluetooth function on the iOS device may be turned off. The Bluetooth device may have been connected to the phone previously. Or the notification function on the iOS device is disabled.

- Solution

1. Check in the **Settings** of the iOS device whether the Bluetooth device has been connected to the mobile phone before. If connection has been established previously, tap **Goodix_ANCS_C** to **Forget This Device**, and redo the scanning, pairing, and bonding.
2. Ensure the notification function of the iOS device is enabled.